(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 2 913 749 A1

# (12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
**02.09.2015 Bulletin 2015/36**

(51) Int Cl.:
***G06F 9/44*** *(2006.01)* ***G06F 3/0481*** *(2013.01)*

(21) Application number: **14167608.0**

(22) Date of filing: **09.05.2014**

(84) Designated Contracting States:
**AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR**
Designated Extension States:
**BA ME**

(30) Priority: **26.02.2014 EP 14156746**

(71) Applicant: **Advanced Digital Broadcast S.A.**
**1292 Geneva (CH)**

(72) Inventors:
- **Kobzda, Piotr**
  **65-119 Zielona Gora (PL)**
- **Pingot, Pawel**
  **65-119 Zielona Gora (PL)**

(74) Representative: **Blonski, Pawel**
**Advanced Digital Broadcast Polska**
**Ul. Trasa Polnocna 16**
**65-119 Zielona Gora (PL)**

## (54) Method and system for focus management in a software application

(57) A method for focus management in a software application, wherein at least a subset of Node objects of said software application forms a hierarchy of Node objects and wherein each Node object of said hierarchy of Node objects of said software application comprises: a first routine that when returning true denotes that the Node object is a focused one; and a second routine that when returning true denotes that the Node object is a focused, or at least one of its descendants is a focused; a routine for handling an incoming event; the method comprising the steps of: providing, for each Node object of said hierarchy of Node objects a third routine, that when returning true denotes that the Node object and all descendants of the Node object, excluding these Node objects for which the third routine returns true, forms a single monofocus area having a single focus root.

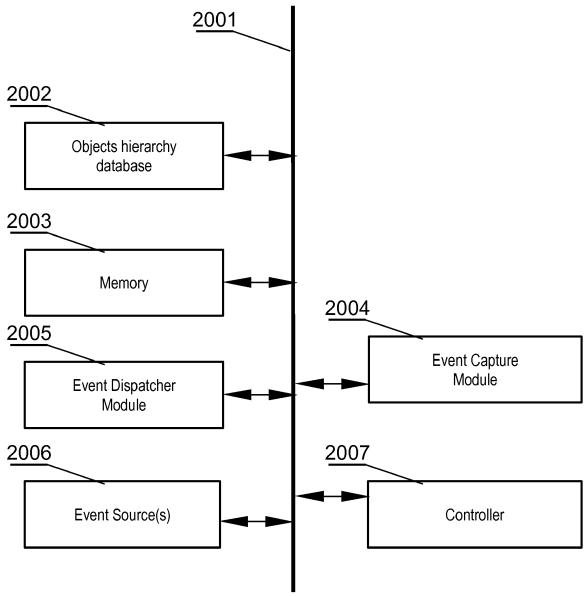


Fig. 20

EP 2 913 749 A1

## Description

### TECHHNICAL FIELD

**[0001]** The present invention relates to a method and system for focus management in a software application. In particular, the invention is applicable in graphical user interface (GUI) software where it is advantageous to have a multiple focus capability.

### BACKGROUND ART

**[0002]** In computing, the focus indicates the component of the graphical user interface which is selected to receive input. Text entered at the keyboard or pasted from a clipboard is sent to the component which has the focus. Moving the focus away from a specific user interface element is known as a blur event in relation to this element. Typically, the focus is withdrawn from an element by giving another element the focus. This means that focus and blur events typically both occur virtually simultaneously, but in relation to different user interface elements, one that gets the focus and one that gets blurred (source: Wikipedia).

**[0003]** In other words, object focus is by its nature an exclusive property of a GUI object - only a single component within components hierarchy may be focused.

**[0004]** In this context there is typically applied a boolean property called "focused" that indicates a focused object, usually complemented with additional boolean property called "has focus". The object having the "focused" property set to true is the exclusively focused object in the objects hierarchy. The "has focus" property is different from "focused" property in that if "has focus" is true it means that a given object or one of his descendants is focused. There is a chain of "has focus"-ed objects till an object that has "is focused" set to true is reached.

**[0005]** This approach is used in so-called event capturing or bubbling techniques.

**[0006]** The main principle of bubbling is that after an event triggers on the deepest possible element, it then triggers on parents in nesting order. The bubbling goes to the top of objects hierarchy. When an event occurs on an object, it will bubble up to the parent object, triggering appropriate handlers. In any case a handler may decide that event is fully processed and stop the bubbling.

**[0007]** In turn capturing is opposite to bubbling and starts at the top object in the hierarchy in order to propagate downwards in the hierarchy structure of the objects tree.

**[0008]** A known prior art patent publication of US 8347215 B2 entitled "Simultaneous input across multiple applications" discloses that one or more users may interact simultaneously with different applications on the same device through an input system such as a touch-sensitive display screen. Simultaneous user input may be detected by a multiple input system and subsequently transmitted to an application using a single transmission frame. An application corresponding to the user input is determined based on an input location of the user input and a location and size of a corresponding application. Simultaneous user inputs are rendered simultaneously by each corresponding application. Each application may further include a secondary event processing thread that runs in parallel to traditional operating system event threads. The secondary event processing thread bypasses sequential processing limits of an operating system event thread, providing substantially simultaneous processing of input.

**[0009]** A drawback of this solution is that different focus managers must be employed. Another disadvantage is that a special secondary event processing thread that bypasses typical event propagation architecture must be applied. Hence it is a significant interference with the legacy systems. Further there is a separate event distribution manager and multithreaded event processing, which is very resource consuming, for example in terms of subsequent resources synchronization.

**[0010]** Another prior art publication of EP0368779 B1 entitled "Method for concurrent data entry and manipulation in multiple applications" discloses methods for inputting common data into a plurality of computer application programs and in particular to methods for automatically and concurrently entering common data into a plurality of computer application programs. Still more particularly, this invention relates to methods which permit the concurrent manipulation of common data which is present within a plurality of computer application programs.

**[0011]** A drawback of this solution is that a special action must be performed by a user on several components and that all the selected components must act in the same manner.

**[0012]** It is sometimes needed to have more than one component that is in focus and may receive events such as keyboard keystrokes notifications. Another advantage of multi focus system is that a single application may run a plurality of submodules that may be unrelated to each other.

**[0013]** Therefore, there exists a problem of efficient implementation of a multi focus system in a software application.

**[0014]** It would be thus desirable to provide an improved method for managing multi focus in a software application.

### SUMMARY OF THE INVENTION

**[0015]** The object of the present invention is a method for focus management in a software application, wherein at least a subset of Node objects of said software application forms a hierarchy of Node objects and wherein each Node object of said hierarchy of Node objects of said software application comprises: a first routine that when returning true denotes that the Node object is a

focused one; and a second routine that when returning true denotes that the Node object is a focused, or at least one of its descendants is a focused; a routine for handling an incoming event; the method comprising the steps of: providing, for each Node object of said hierarchy of Node objects a third routine, that when returning true denotes that the Node object and all descendants of the Node object, excluding these Node objects for which the third routine returns true, forms a single monofocus area having a single focus root.

**[0016]** Preferably, the first routine is a isFocused() routine, the second routine is an hasFocus() routine and the third routine is a isFocusRoot() routine.

**[0017]** Preferably, the routine for handling an incoming event is such that it guarantees that the event being dispatched will first be processed by focused focus roots Node objects before handling of the event by focused child Node object.

**[0018]** Preferably, the routine for handling an incoming event comprises the steps of: handling the event by a focus root; dispatching the event to focus roots; and handling the event by a focus owner.

**[0019]** Preferably, the handling the event by a focus root comprises the steps of: checking whether the Node object is a focused focus root and then handling of the event.

**[0020]** Preferably, the dispatching the event to focus roots comprises the steps of: iterating in a loop, in reverse z-order, all children nodes of the Node object that has the focus, except for child node that is focused or has descendant a focus owner child of the monofocus subsection comprising the given Node object; dispatching the event to the Node objects that are found to be a focused focus roots.

**[0021]** Preferably, the handling the event by a focus owner comprises the steps of: dispatching of the event to the focused child node or, if the given Node object is focused and is not focus root, handling the event by the Node object itself.

**[0022]** Preferably, the method further comprises a step of: providing for each Node object a fourth routine unsetting the property referred to by said third routine wherein the fourth routine takes a preserve focus parameter as an input and is configured to clear focus on all conflicting nodes while preserving the focus where it was before the fourth routine execution.

**[0023]** Preferably, the conflict of focus exists with a parent monofocus subsection, to which the monofocus subsection under consideration is to be joined.

**[0024]** Another object of the present invention is a computer program comprising program code means for performing all the steps of the method according to the method of the present invention when said program is run on a computer.

**[0025]** Another object of the present invention is a computer readable medium storing computer-executable instructions performing all the steps of the method according to the present invention when executed on a computer.

**[0026]** Yet another object of the present invention is a system for focus management the system comprising: a data bus communicatively coupled to a memory; a controller communicatively coupled to the system bus; the system further comprising: an events sources controller for monitoring hardware event sources in order to receive events and notify the events, by means of the event capture module to an event dispatcher module; wherein the event dispatcher module is configured to, under the supervision of the main controller, manage dispatching events to respective objects, in a given hierarchy of application objects, that are stored in an objects hierarchy database wherein each object in the objects hierarchy database is a Node object according to the aforementioned method of the present invention.

**[0027]** Preferably, the event dispatcher module is configured to operate according to a method of the present invention.

**[0028]** Preferably, the main controller is configured to execute one or more software applications stored in the memory.

**[0029]** Preferably, the applications preferably comprise a graphical user interface.

## BRIEF DESCRIPTION OF THE DRAWINGS

**[0030]** The present invention is shown by means of exemplary embodiments on a drawing, in which:

Fig. 1A and 1B present a simple objects hierarchy example;

Fig. 2A and Fig. 2B present a relation between hasFocus() and isFocused() in a classic focus approach;

Fig. 3A and Fig. 3B present focus approach according to the present invention;

Fig. 4A and Fig. 4B present a different example of the focus approach according to the present invention;

Fig. 5A and Fig. 5B present a first clear focus scenario;

Fig. 6A and Fig. 6B present a second clear focus scenario;

Fig. 7A and Fig. 7B present a third clear focus scenario;

Fig. 8A and Fig. 8B present a first set focus root scenario;

Fig. 9A and Fig. 9B present a second set focus root scenario;

Fig. 10A and Fig. 10B present a first unset focus root scenario;

Fig. 11A and Fig. 11B present a second unset focus root scenario;

Fig. 12A and Fig. 12B present a first unset focus root scenario with focus preservation;

Fig. 13A and Fig. 13B present a second unset focus root scenario with focus preservation;

Fig. 14 presents a simplified UML class diagram of an exemplary tree node provided in order to explain details of the present invention;

Fig. 15 and Fig. 16 shows top level view of event propagation method according to the present invention;

Fig. 17, Fig. 18 and Fig. 19 present details of an event propagation method; and

Fig. 20 presents a system according to the present invention.

## DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

### NOTATION AND NOMENCLATURE

**[0031]** Some portions of the detailed description which follows are presented in terms of data processing procedures, steps or other symbolic representations of operations on data bits that can be performed on computer memory. Therefore, a computer executes such logical steps thus requiring physical manipulations of physical quantities.

**[0032]** Usually these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated in a computer system. For reasons of common usage, these signals are referred to as bits, packets, messages, values, elements, symbols, characters, terms, numbers, or the like.

**[0033]** Additionally, all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Terms such as "processing" or "creating" or "transferring" or "executing" or "determining" or "detecting" or "obtaining" or "selecting" or "calculating" or "generating" or the like, refer to the action and processes of a computer system that manipulates and transforms data represented as physical (electronic) quantities within the computer's registers and memories into other data similarly represented as physical quantities within the memories or registers or other such information storage.

**[0034]** Fig. 1A presents a simple objects hierarchy example. The example includes a main window object 101 (such as JFrame in Java), which is a top object in the objects tree. The window 101 comprises two child objects 102 and 104 that are in this case a panel (such as JPanel in Java) and a dialog window (such as JDialog in Java). Each first level child object comprises a second level child object 103, 105 that in this case are a panel and a button object (such as JButton in Java). Similarly, in case of the 105 panel a third level child object in present i.e. a button 106. In case the button 106 is focused, no other GUI object may be focused as well.

**[0035]** In order to be presented on a display screen, every GUI component must be part of a containment hierarchy. A containment hierarchy is a tree of components that has a top-level container as its root. In this case the component 101 is a root. Each root component has a content pane that comprises (directly or indirectly) the visible components in that top-level container's GUI.

**[0036]** Fig. 1B presents the same components positioned in a parent-child relationship in order to better visually present their relationships in the graphical user interface.

**[0037]** It is the aim of the present invention to, for example, facilitate focus for example on the component 103 as well as on the component 106 at the sime time.

**[0038]** Figures 2A to 13B refer to different uses and exemplary, non-limitative cases of requesting focus, clearing focus, setting focus root and unsetting focus root.

**[0039]** The notation in these figures is such that a horizontal dash in the circle denotes that the hasFocus() property of the given node is set to true. Further, a vertical dash in the circle denotes that the isFocused() property of the given node is set to true. Further, a circle drawn with a dashed line around a given node's circle denotes that the isFocusRoot() property of the given node of the hierarchy is set to true.

**[0040]** Fig. 2A and Fig. 2B present a relation between hasFocus() and isFocused() properties in a classic focus single focus approach. In Fig. 2A a focused element 203 has the hasFocus() and isFocused() properties set to true while the elements 202 and 201 only have hasFocus() property set to true while isFocused() property remains set to false. This is because one of their descendants is focused.

**[0041]** In case focus is requested by a 204 element, the arrangement shown in Fig. 2B will be the result. The 204 element does not have focus and its hasFocus() and isFocused() properties are both set to false. The previously set hasFocus() property of the 202 component is not also set to false as none of its descendants is focused.

**[0042]** On the other hand a focused element 204 has the hasFocus() and isFocused() properties set to true while the elements 205 and 201 only have hasFocus() property set to true while isFocused() property remains set to false. This is again because one of their descendants is focused.

**[0043]** The present invention introduces to the ar-

rangement shown in Fig. 2A and Fig. 2B another boolean property of node (sometimes referred to as an element or a component) called "focus root", indicated as isFocusRoot() routine.

**[0044]** Consequently, the key node actions are requestFocus(), clearFocus(), setFocusRoot(), unsetFocusRoot() and their tasks are self explanatory given their names.

**[0045]** Further, the nodes states (properties) are as follows: isFocused(), hasFocus() and isFocusRoot().

**[0046]** Fig. 3A and Fig. 3B present focus approach according to the present invention. In Fig. 3A a focused element 303 has the hasFocus() and isFocused() properties set to true while the element 302 has the isFocusRoot() property set to true and the hasFocus() property set to true (the isFocused() property is set to false) and the element 301 only has hasFocus() property set to true while the isFocused() property remains set to false. This is because one of their descendants, in particular the 303 node, is focused.

**[0047]** It is to be noted that the isFocusRoot() property may be set to true on any node of the objects hierarchy and that there may be multiple objects having isFocusRoot() property set to true.

**[0048]** In case focus is requested by a 304 element, the arrangement shown in Fig. 3B will be the result. The 304 element does not have focus and its hasFocus() and isFocused() properties are initially both set to false. The configuration of the 301, 302 and 303 elements remains as the isFocusRoot() property remains set to true and the node 303 having isFocused() property set to true is not affected by focus change.

**[0049]** On the other hand a focused element 304 has the hasFocus() and isFocused() properties set to true while the elements 305 and 301 only have hasFocus() property set to true while isFocused() property remains set to false. This is again because one of their descendants is focused.

**[0050]** Fig. 4A and Fig. 4B present another example of the focus approach according to the present invention. It relates to requesting focus in case when there are two focused objects 403 and 404, one of which is under focus root node.

**[0051]** In Fig. 4A a focused element 403 has the hasFocus() and isFocused() properties set to true while the element 402 has the isFocusRoot() property set to true and the hasFocus() property set to true (the isFocused() property is set to false) and the element 401 only has hasFocus() property set to true while the isFocused() property remains set to false. This is because one of their descendants, 403 node, is focused.

**[0052]** Additionally, the element 404 is a focused element i.e. both hasFocus() and isFocused() properties are set to true while the isFocusRoot() property is set to false. Consequently, the 405 and 401 nodes have the hasFocus() property set to true and the isFocused() property set to false and further the respective isFocusRoot() property set to false.

**[0053]** Setting the isFocusRoot() property to true creates in a sense a subrange of hierarchy nodes in which a single focus arrangement exists. For example, a single focus arrangement exists for nodes group 407 and simultaneously a single focus arrangement exists for nodes group 408. Such approach creates as a result multiple monofocus subsections 407, 408 in the objects tree hierarchy.

**[0054]** More generally, a isFocusRoot() property set to true denotes that the Node object and all descendants of the Node object, except for other descendant Node objects having the isFocusRoot() property set to true, form a single monofocus area having a single focus root. In other words, the aforementioned single monofocus area excludes these Node objects for which the isFocusRoot() property is set to true.

**[0055]** Owing to this approach an exemplary focus change will be defined as shown in Fig. 4B.

**[0056]** In view of the above, when focus is requested for node 406, which is within the group 408, the focus will be cleared from the 403 node and set on the 406 node. This will not affect nodes present in the group 407 as shown in Fig. 4B.

**[0057]** Fig. 5A and Fig. 5B present a first clear focus scenario. In Fig. 5A a focused element 503 has the hasFocus() and isFocused() properties set to true while the element 502 has the isFocusRoot() property set to true and the hasFocus() property set to true (the isFocused() property is set to false) and the element 501 only has hasFocus() property set to true while the isFocused() property remains set to false. This is because one of their descendants is focused.

**[0058]** Clearing focus in the arrangement of Fig. 5A will result in a state of Fig. 5B wherein the 502 element remains with its isFocusRoot() property set to true end the hasFocus() and isFocused() properties set to false while for the nodes 501 and 503 all of the three properties are set to false. Consequently, there is not any focused node in the objects hierarchy.

**[0059]** Fig. 6A and Fig. 6B present a second clear focus scenario. The scenario relates to clearing focus in case when there are two focused objects 603, 604 one of which is under a focus root node 602.

**[0060]** In Fig. 6A a focused element 603 has the hasFocus() and the isFocused() properties set to true while the element 602 has the respective isFocusRoot() property set to true and the hasFocus() property set to true (the isFocused() property is set to false) and the element 601 only has the hasFocus() property set to true while the isFocused() property remains set to false. This is because one of their descendants, the 603 node, is a focused node.

**[0061]** Additionally, the element 604 is a focused element i.e. the hasFocus() and the isFocused() properties are set to true while the isFocusRoot() property is set to false. Consequently, the aforementioned 605 and 601 nodes have hasFocus() property set to true and isFocused() property set to false and isFocusRoot() property

set to false.

**[0062]** As already explained in such a case single focus arrangement exists for nodes 607 and simultaneously a single focus arrangement exists for nodes 608. Owing to this approach an exemplary focus clear will be defined as shown in Fig. 6B.

**[0063]** In view of the above, when focus is cleared for the node 603, which is within the group 608, the focus set on the 606 node will not be affected regarding focus.

**[0064]** Fig. 7A and Fig. 7B present a third clear focus scenario. In Fig. 7A a focused element 703 has the hasFocus() and isFocused() properties set to true while the element 702 has the isFocusRoot() property set to true and the hasFocus() property set to true (the isFocused() property is set to false) and the element 701 only has hasFocus() property set to true while the isFocused() property remains set to false. This is because one of their descendants is focused.

**[0065]** Additionally, the element 704 is a focused element i.e. the hasFocus() and the isFocused() properties are set to true while the isFocusRoot() property is set to false. Consequently, the 705 and 701 nodes have the hasFocus() property set to true and the isFocused() property set to false and the respective isFocusRoot() property set to false.

**[0066]** As already explained in such a case single focus arrangement exists for nodes 707 and simultaneously a single focus arrangement exists for nodes 708. Owing to this approach an exemplary outcome of a focus clear action will be defined as shown in Fig. 7B.

**[0067]** In view of the above detailed explanation, when focus is cleared for node 704, which is within the nodes group 707, the isFocused() property and the hasFocus() property are both cleared for the node 704 while the hasFocus() property is cleared from the 705 node i.e. set to false. The hasFocus() property remains set to true for the 701 node as one of its descendants, the 703 node is currently focused.

**[0068]** The following figures 8A - 9B present different examples of setFocusRoot() scenarios moving or setting focus on given nodes. It is to be noted that current focus within the hierarchy tree is never affected by setting focus root on an additional node.

**[0069]** Fig. 8A and Fig. 8B present a first set focus root scenario. In Fig. 8A a focused element 803 has the hasFocus() and the isFocused() properties set to true while the element 802 has the hasFocus() property set to true (the isFocused() property is set to false and isFocusRoot() property is set to false) and the element 801 only has the hasFocus() property set to true while the isFocused() and the isFocusRoot() properties remain set to false. This is because one of their descendants, namely the 803 node, is focused.

**[0070]** In this state there may be only one focused object as all nodes form a single set of nodes because there is not any focus root object having the isFocusRoot() property set to true.

**[0071]** In this scenario focus root is requested for node 802, which results in an arrangement shown in Fig. 8B. In this case the only element that changes in comparison to Fig. 8A is that the isFocusRoot() property is set to true for the 802 node. This results in that two monofocus subsections are created within the objects hierarchy. Namely these monofocus subsections are outlined as 804 and 805 subsections.

**[0072]** Fig. 9A and Fig. 9B present a second set focus root scenario. The arrangement of nodes 901 - 905 in Fig. 9A corresponds to the state as shown in Fig. 8B. The state is further modified by setting another focus root on node 906, for which the isFocusRoot() property is set to true. This results in that the two monofocus subsections are modified into three monofocus areas created within the same objects hierarchy. Namely these monofocus subsections are outlined as 907, 908 and 909 subsections.

**[0073]** Figure 10A and Figure 10B present a first unset focus root scenario. If focus ownership, within focus root subtree, does not conflict with classic single focus semantics, then it may optionally be preserved after unsetting a focus root. Normally, however unsetFocusRoot() always invokes a clearFocus() routine.

**[0074]** In the optional arrangement there may be a property present in the unsetFocusRoot() routine such as preserveFocus (Boolean) which would indicate that all conflicting nodes, in terms of focus, are cleared while preserving the focus where it was before unsetFocusRoot() execution.

**[0075]** In Fig. 10A a focused element 1003 has the hasFocus() and isFocused() properties set to true while the element 1002 has the hasFocus() property and isFocusRoot() property set to true (the isFocused() property is set to false) and the element 1001 only has hasFocus() property set to true while the isFocused() and isFocusRoot() properties remain set to false. This is because one of their descendants is focused.

**[0076]** In such state of the objects tree the unsetFocusRoot() routine is executed resulting in the Fig. 10B state of the objects hierarchy. In such state, all three properties i.e. hasFocus(), isFocused() and isFocusRoot() are set to false for all objects. The properties previously set to true for objects 1001 - 1003 are now set to false because focus root is removed and the focus is cleared under 1002 node of the hierarchy.

**[0077]** Fig. 11A and Fig. 11B present a second unset focus root scenario. In this scenario focus ownership within focus root subtree is in conflict with classic focus semantics. In such case the clearing must be executed (i.e. clearFocus() is performed) before unsetting focus root.

**[0078]** In Fig. 11A a focused element 1103 has the hasFocus() and isFocused() properties set to true, while the element 1102 has the isFocusRoot() property set to true and the hasFocus() property set to true (the isFocused() property is set to false) and the element 1101 only has hasFocus() property set to true, while the isFocused() property remains set to false. This is because

one of their 1001, 1002 descendants 1003 is focused.

**[0079]** Further, the 1104 node has the hasFocus() and isFocused() properties set to true while the elements 1105 and 1101 only have hasFocus() property set to true while isFocused() property remains set to false. This is again because one of their descendants is focused.

**[0080]** If in such a state of the objects hierarchy, a request to unset the focus root for the 1102 object is invoked, the output state as shown in Fig. 11B will be the final result. The focus root 1102 is unset by setting the isFocusRoot() and the hasFocus() properties to false and prior to that, clearing the focus from the 1103 node by setting the hasFocus() and the isFocused() properties to false. Consequently, there is not any focus root present in the Fig. 11B objects hierarchy and all nodes operate under a classic monofocus arrangement until a focus root is established in the objects hierarchy.

**[0081]** Fig. 12A and Fig. 12B present a first unset focus root scenario with focus preservation. This is an optional approach as previously indicated. The Fig. 12A arrangement matches Fig. 10A. The difference lies however in the output arrangement of Fig. 12B in comparison to Fig. 10B.

**[0082]** By using a special preserveFocus option set to true (preferably a parameter of a boolean type), the user may indicate that, at a time when the unsetFocusRoot() routine is invoked, the focus within this monofocus subsection 1204 shall be preserved.

**[0083]** Therefore, as clearly shown in Fig. 12B, only the 1202 node's isFocusRoot() property is set to false since there are no conflicts of focus with a parent monofocus subsection 1205, to which the monofocus subsection 1204 under consideration will be joined.

**[0084]** Fig. 13A and Fig. 13B present a second unset focus root scenario with focus preservation. This is an optional approach as previously indicated. The Fig. 12A arrangement matches Fig. 11A. The difference lies however in the output arrangement of Fig. 13B in comparison to Fig. 10B.

**[0085]** By using the preserveFocus option set to true the user may indicate that at a time when the unsetFocusRoot() routine is invoked the focus within this monofocus subsection shall be preserved. Therefore, as shown in Fig. 13B, the 1304 focus present in the parent monofocus subsection 1305 shall be cleared in order to preserve focus unchanged on the node focused in the child monofocus subsection.

**[0086]** In case there are multiple levels of objects hierarchy, the focus preservation method only evaluates a potential conflict with its direct parent monofocus subsection.

**[0087]** Fig. 14 presents a simplified UML class diagram of an exemplary object tree node provided in order to explain details of the present invention. The exemplary Node class comprises the attributes of the class 1401 that are representative of the state of the Node object, and methods (operations of the class), marked as 1402 that form functional routines of the Node.

**[0088]** The focused 1403 and the focusRoot 1404 attributes are preferably both boolean flags (or any other suitable variable type capable of assuming one of two states) that indicate current focused state and focus root state of the given Node object.

**[0089]** These two attributes both preferably have read access methods, isFocused() 1405 and isFocusRoot() 1406 respectively, which are used to retrieve the given state of the Node object. There is also a hasFocus() 1407 method that returns true if the Node object is focused, or any other node within subtree rooted by the Node object is focused (a descendant Node object of the Node object in question).

**[0090]** The mutators (In computer science, a mutator method is a method used to control changes to a variable. They are also widely known as setter methods. Often a setter is accompanied by a getter (also known as an accessor), which returns the value of the private member variable) of the focused attribute are requestFocus() and clearFocus() methods. The requestFocus() method is used to enable (set to true) the focused attribute, while clearFocus() disables (sets to false) the respective attribute. Both these methods correctly maintain a single focus within monofocus subsection comprising the Node object, and take care of other focus related states of all nodes of the tree.

**[0091]** Similarily, mutators of the focusRoot attribute are setFocusRoot() and unsetFocusRoot() methods, where the setFocusRoot() method is used to enable the focusRoot attribute. The unsetFocusRoot() provides a means for disabling the focusRoot attribute with a preserveFocus option, that allows to make a decision on how to resolve potential single focus problems within a resulting monofocus subsection comprising the Node object that will effectively be created.

**[0092]** The childrenNodes attribute holds direct children nodes of the Node object that all, starting from the root Node object, forms the nodes tree, while, in a preferred embodiment of the present invention, the parent attribute holds a direct reference to a parent Node object (except for the root Node object that does not have a parent).

**[0093]** There preferably also exists a special focusedChild attribute that indicates the node's child node which holds (or is) a focus owner within a given monofocus subsection comprising the Node object. The focusedChild attribute is maintained by means of requestFocus() / clearFocus() and setFocusRoot()/unsetFocusRoot() methods.

**[0094]** The Node has also dispatchEvent() method that receives an event as input and implements the event's propagation algorithm within the nodes tree. The dispatchEvent() method decides which Node objects, and according to which order, will be requested for the event handling, and then passes the event to this Node object's handleEvent() method. The handleEvent() method is a node specific implementation which is responsible for event "consumption", which ends the event propagation,

or just ignore the event, which causes handling an event by next Node object in event propagation order.

**[0095]** Fig. 15 and Fig. 16 show event propagation method according to the present invention. At step 1501 in Fig. 15 the system (such as Java JRE or similar) propagates an event to the nodes tree of a running software application. Subsequently, at step 1502, the root node object takes over handling of the received event. This event handling has been depicted in more details in Fig. 16. The last step of the process shown in Fig. 15 is the end of the notification process.

**[0096]** Fig. 16 presents a top level of the process defined by event propagation algorithm by each Node object.

**[0097]** The step 1601 forms the Node object's dispatchEvent() method entry point that takes as input an event that is being dispatched (such event may for example be received from a keyboard and passed to a running application by a runtime environment such as Java Runtime Environent or the like).

**[0098]** The event propagation process comprises execution of steps 1602, 1603, and 1604 in the given order, which are preferably Node object's private subroutines called handleByFocusRoot(), dispatchToFocusRoots() and handleByFocusOwner(). Each of these subroutines, similarly to the dispatchEvent() method, takes an event as parameter and returns true if the event has been processed (routine has "consumed" the event), or otherwise returns false. The first subroutine that has processed the event causes termination of the dispatchEvent() routine at step 1606 (returns true). If none of these subroutines has consumed an event, then the routine ends at step 1605 (returns false).

**[0099]** The execution order of subroutines invoked at steps 1602, 1603, and 1604 guarantees that the event being dispatched will first be processed by focused focus roots subtrees before handling of the event by focused child Node object. This is important feature of the present invention that enables extendability of possibly complex hierarchies by addition of independent focus root nodes that can handle an events before other nodes in the tree without affecting of the focus states of existing tree nodes.

**[0100]** For example, such auxiliary focused focus root nodes can serve as the modal overlay of current graphical user interface components, fully decoupling the events handling logic from the other components in the objects hierarchy tree.

**[0101]** Fig. 17 presents the aforementioned handleByFocusRoot() subroutine, which is a first step of the event propagation routine presented in Fig. 16 as step 1602.

**[0102]** The step 1701 forms an entry point of the subroutine that takes as input an event that is being processed. The routine comprises checking at steps 1702, and 1703 whether the Node object is a focused focus root and if so, handling of an event at step 1704 is performed. If that event is handled, then the subroutine ends at step 1705 (returns true). Otherwise the subroutine ends at step 1706 (returns false).

**[0103]** Fig. 18 presents the dispatchToFocusRoots() subroutine which is a second step of the aforementioned event propagation routine presented in Fig. 16 as step 1603.

**[0104]** The step 1801 forms an entry point of the subroutine that takes as input an event that is being processed. The routine comprises of a loop that iterates, in reverse z-order, all children nodes of the Node object that has the focus, but are not a focus owner child of the monofocus subsection comprising the given Node object. The loop is initialized at step 1802, and a loop end condition is checked at step 1803.

**[0105]** The child node for event processing is determined at step 1804. Conditions that are mandatory for a child node are checked at steps 1805 and 1806 which ensure that a subtree rooted by this child Node contains at least one focused monofocus subtree and is not currently focused child of the Node (processing of which is postponed to handleByFocusOwner() subroutine). And then, if the child node meets the conditions, an event is dispatched to that node (that is, dispatchEvent() method is invoked on it) at step 1807. If an event has been consumed by the child node, then the routine ends at step 1808 (returns true). Otherwise, the subroutine continues the loop from step 1803, and, if there are no more children nodes to process, ends at step 1809 (returns false).

**[0106]** Fig. 19 presents the handleByFocusOwner() subroutine which is a third step of the aforementioned event propagation routine presented in Fig. 16 as step 1604.

**[0107]** Step 1901 forms an entry point of the subroutine that takes as input an event that is being processed. The routine comprises dispatching of the received event to the focused child node (if any) or, if the given Node object is focused, handling the event by the Node object itself. Checks performed at steps 1902 and 1903 ensure that the event has not been already processed by a focused focus root node (by handleByFocusRoot() subroutine), and if so then the event is being handled by the Node object at step 1904.

**[0108]** Alternatively, if there is a focused child set for the Node object, which is ensured at step 1905, then an event is dispatched to that child node. If an event is either handled by the Node object, or consumed by focused child node, then the routine ends at step 1908 (returns true). Otherwise the subroutine ends at step 1907 (returns false).

**[0109]** Fig. 20 presents the system according to the present invention. The system may be realized using dedicated components or custom made FPGA (Field Programmable Gate Array) or ASIC (Application Specific Integrated Circuit) circuits or as a combination of software and hardware. The system comprises a data bus 2001 (such as for example I2C) communicatively coupled to a memory 2002. Additionally, other components of the system are communicatively coupled to the system bus 2001 so that they may be managed by a controller 2003.

**[0110]** The system comprises further events sources

controller 2006 for monitoring hardware event sources such as a computer keyboard or a mouse in order to receive events and notify them, by means of the event capture module 2004 to an event dispatcher module 2005. The event dispatcher module 2005, under the supervision of the main controller 2003, manages dispatching events to respective objects, in a given hierarchy of application objects, that are stored in an objects hierarchy database 2002. Each object in the objects hierarchy database 2002 is a Node object according to definition of Fig. 14 and the event dispatcher module 2005 operates according to a method defined in Fig. 15 to Fig. 19.

**[0111]** The main controller 2003 may also be configured to execute one or more software applications stored in the memory 2002. The applications preferably comprise graphical user interface.

**[0112]** While all exemplary embodiments depict a three level objects structure, one skilled in the are will easily apply the same approach to objects structures having fever or more logical levels of objects.

**[0113]** The multiple focus management using focus roots allows for simple, resources effective multifocus management. There is not required a window manager and returning to a previously focused object is very simple. Further there is not required a sophisticated focus manager of many focus managers running in parallel. Event processing is executed at the level of nodes that control consuming.

**[0114]** Additionally, it is to be noted that it is allowable that all objects are focused objects.

**[0115]** The distribution of events in branches which are focus roots may be executed from the newest to the oldest object.

**[0116]** It can be easily recognized, by one skilled in the art, that the aforementioned method for determining an index of an object in a sequence of objects may be performed and/or controlled by one or more computer programs. Such computer programs are typically executed by utilizing the computing resources of the device. The computer programs can be stored in a non-volatile memory, for example a flash memory or in a volatile memory (or otherwise a non-transitory computer readable medium), for example RAM and are executed by the processing unit. These memories are exemplary recording media for storing computer programs comprising computer-executable instructions performing all the steps of the computer-implemented method according the technical concept presented herein.

**[0117]** While the invention presented herein has been depicted, described, and has been defined with reference to particular preferred embodiments, such references and examples of implementation in the foregoing specification do not imply any limitation on the invention. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader scope of the technical concept. The presented preferred embodiments are exemplary only, and are not exhaustive of the scope of the technical concept presented herein.

**[0118]** Accordingly, the scope of protection is not limited to the preferred embodiments described in the specification, but is only limited by the claims that follow.

**[0119]** In addition, any combination of the appended claims is envisaged in the present application.

## Claims

1. A method for focus management in a software application, wherein at least a subset of Node objects of said software application forms a hierarchy of Node objects and wherein each Node object of said hierarchy of Node objects of said software application comprises:

   • a first routine (1405) that when returning true denotes that the Node object is a focused one; and
   • a second routine(1407) that when returning true denotes that the Node object is a focused, or at least one of its descendants is a focused;
   • a routine for handling an incoming event;

   the method being **characterized in that** it comprises the steps of:

   • providing, for each Node object of said hierarchy of Node objects a third routine (1406), that when returning true denotes that the Node object and all descendants of the Node object, excluding these Node objects for which the third routine returns true, forms a single monofocus area (805, 907, 909) having a single focus root.

2. The method according to claim 1 **characterized in that** the first routine is a isFocused() routine, the second routine is an hasFocus() routine and the third routine is a isFocusRoot() routine.

3. The method according to claim 1 **characterized in that** the routine for handling an incoming event (1601) is such that it guarantees that the event being dispatched will first be processed by focused focus roots Node objects before handling of the event by focused child Node object.

4. The method according to claim 3 **characterized in that** the routine for handling an incoming event (1601) comprises the steps of:

   • handling the event by a focus root;
   • dispatching the event to focus roots; and
   • handling the event by a focus owner.

5. The method according to claim 4 **characterized in that** the handling the event by a focus root comprises

the steps of:

- checking (1702, 1703) whether the Node object is a focused focus root and then:
- handling of the event (1704).

**6.** The method according to claim 4 **characterized in that** the dispatching the event to focus roots comprises the steps of:

- iterating in a loop, in reverse z-order, all children nodes of the Node object that has the focus, except for child node that is focused or has descendant a focus owner child of the monofocus subsection comprising the given Node object;
- dispatching the event to the Node objects that are found to be a focused focus roots.

**7.** The method according to claim 4 **characterized in that** the handling the event by a focus owner comprises of the step:

- dispatching of the event to the focused child node (1906) or, if the given Node object is focused and is not focus root, handling the event by the Node object itself (1904).

**8.** The method according to claim 1 **characterized in that** it further comprises a step of:

- providing for each Node object a fourth routine unsetting the property referred to by said third routine wherein the fourth routine takes a preserve focus parameter as an input and is configured to:
  - clear focus on all conflicting nodes while preserving the focus where it was before the fourth routine execution.

**9.** The method according to claim 8 **characterized in that** the conflict of focus exists with a parent monofocus subsection (1205), to which the monofocus subsection (1204) under consideration is to be joined.

**10.** A computer program comprising program code means for performing all the steps of the method according to claim 1 when said program is run on a computer.

**11.** A computer readable medium storing computer-executable instructions performing all the steps of the method according to claim 1 when executed on a computer.

**12.** A system for focus management the system comprising:

- a data bus (2001) communicatively coupled to a memory (2002);
- a controller (2003) communicatively coupled to the system bus (2001);

the system being **characterized in that** it further comprises:

- an events sources controller (2006) for monitoring hardware event sources in order to receive events and notify the events, by means of the event capture module (2004) to an event dispatcher module (2005);
  - wherein the event dispatcher module (2005) is configured to, under the supervision of the main controller (2003), manage dispatching events to respective objects, in a given hierarchy of application objects, that are stored in an objects hierarchy database (2002)
  - wherein each object in the objects hierarchy database (2002) is a Node object according to claim 1.

**13.** The system according to claim 12 **characterized in that** the event dispatcher module (2005) is configured to operate according to a method defined in claim 4.

**14.** The system according to claim 12 **characterized in that** the main controller (2003) is configured to execute one or more software applications stored in the memory (2002).

**15.** The system according to claim 14 **characterized in that** the applications preferably comprise graphical user interface.

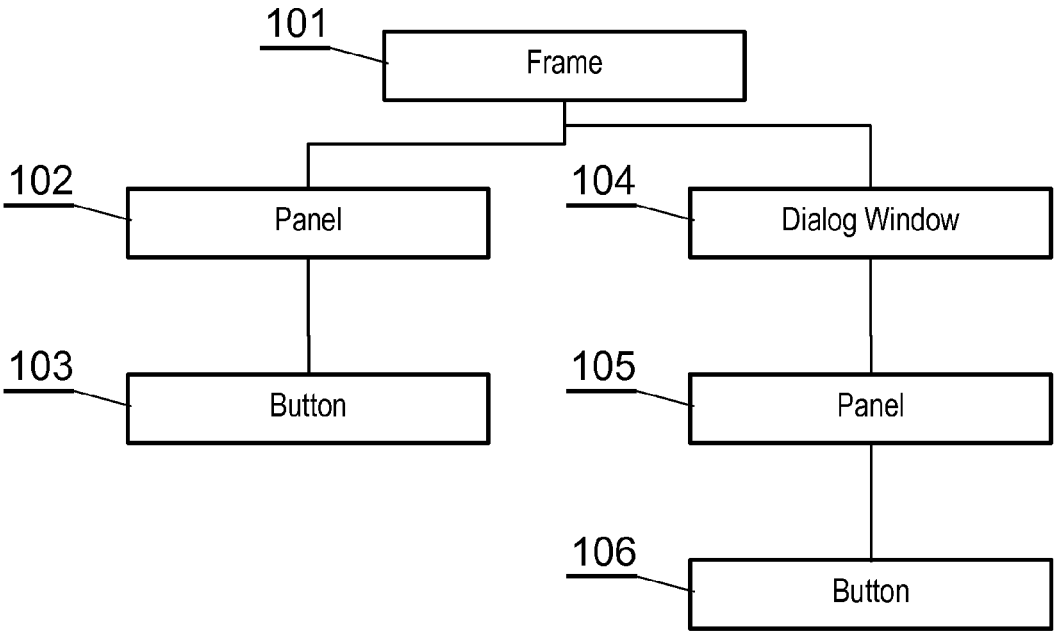


Fig. 1A
(Prior Art)

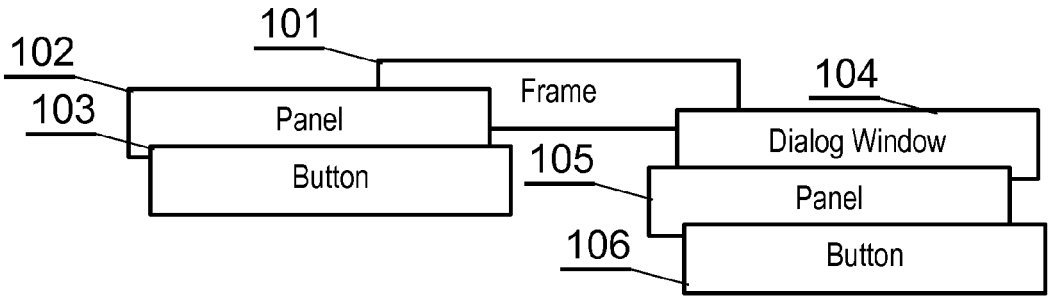


Fig. 1B
(Prior Art)

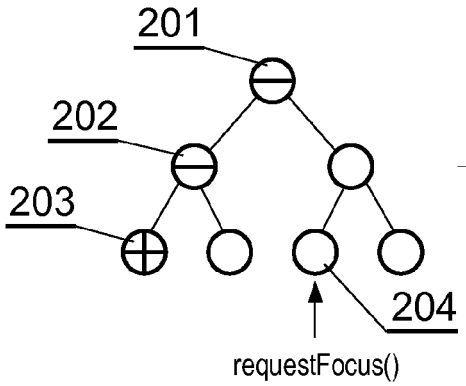


Fig. 2A

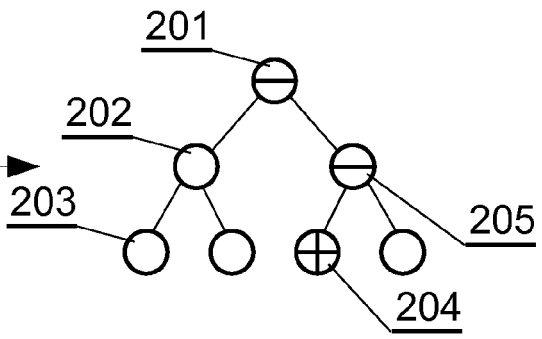


Fig. 2B

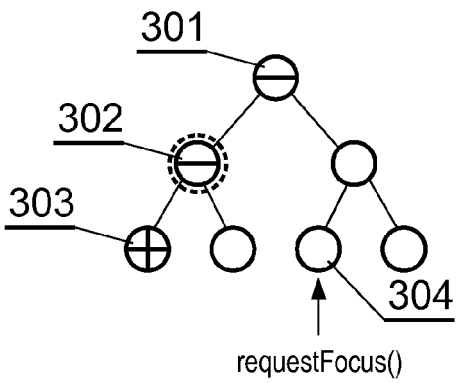


Fig. 3A

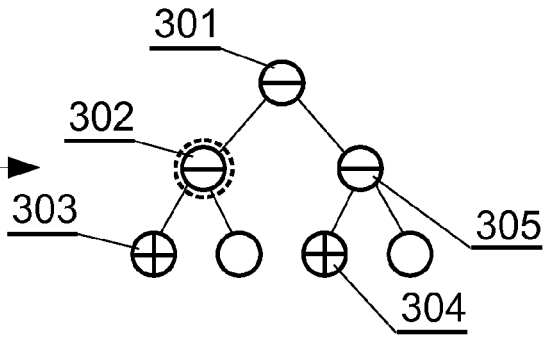


Fig. 3B

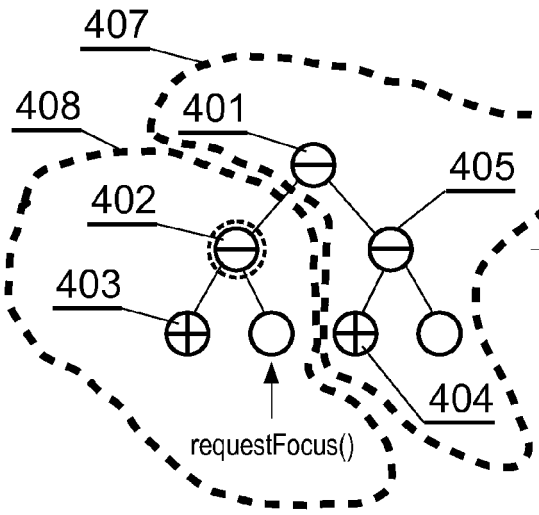


Fig. 4A

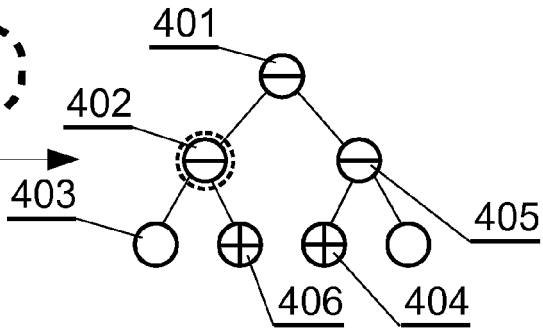


Fig. 4B

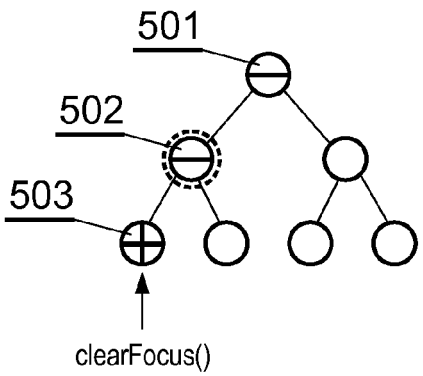


Fig. 5A

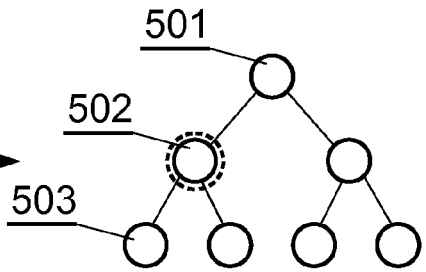


Fig. 5B

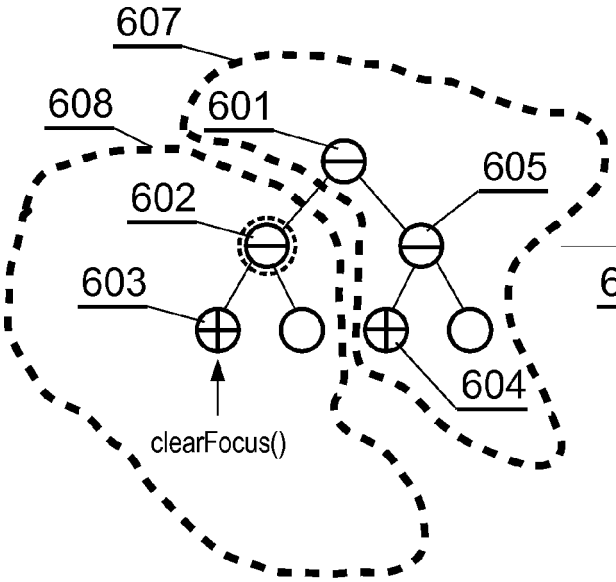


Fig. 6A

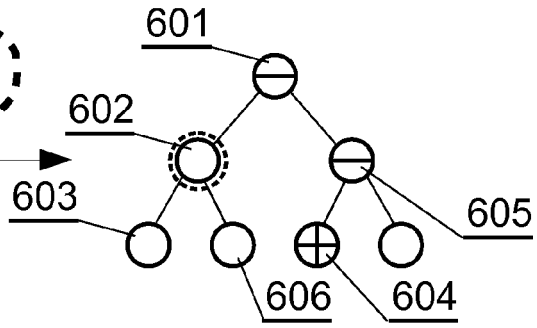


Fig. 6B

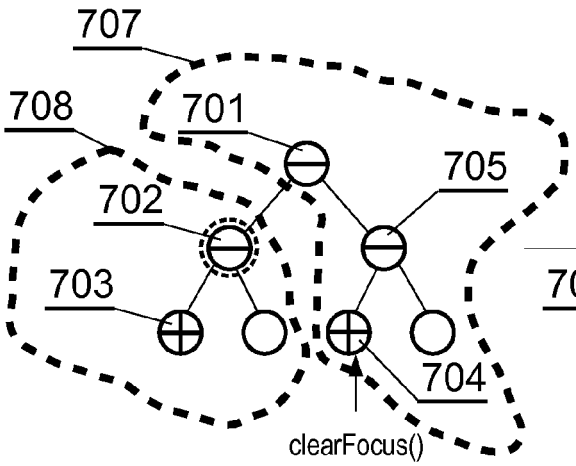


Fig. 7A

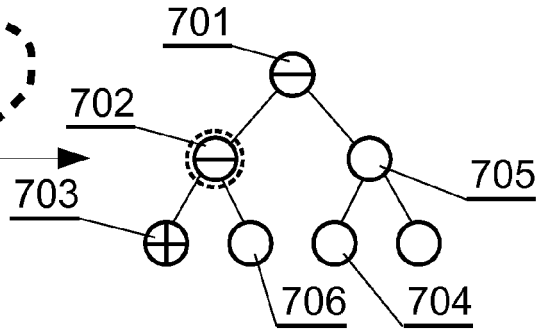


Fig. 7B

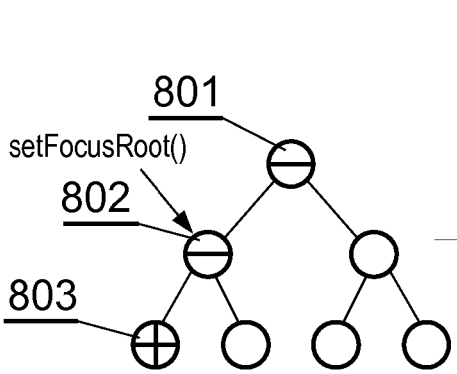


Fig. 8A

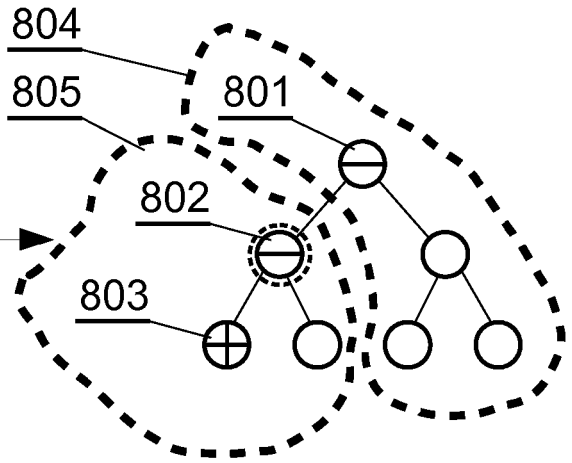


Fig. 8B

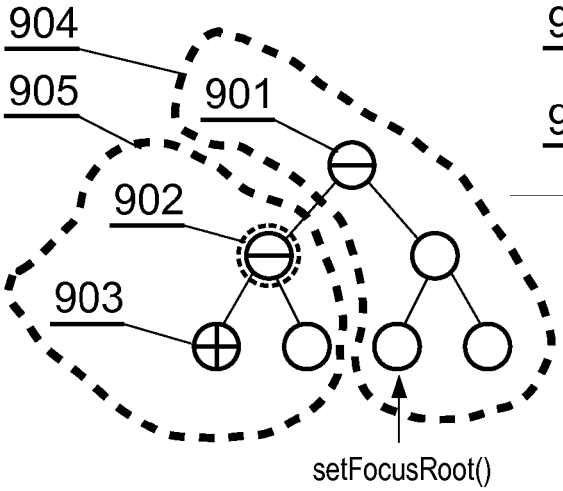


Fig. 9A

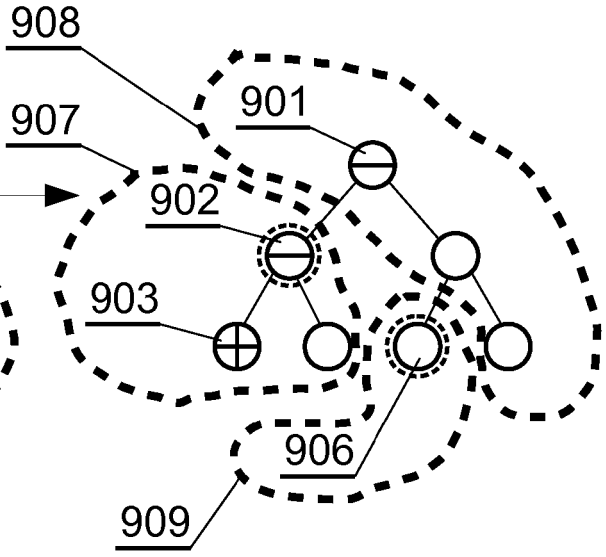


Fig. 9B

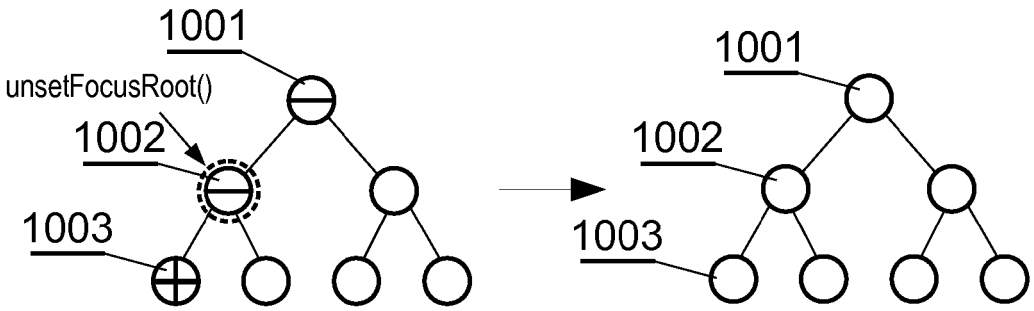


Fig. 10A




Fig. 10B

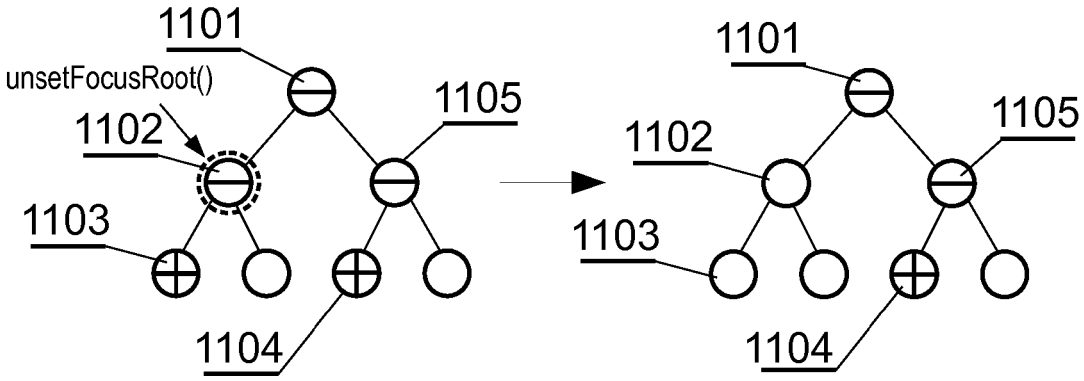


Fig. 11A




Fig. 11B

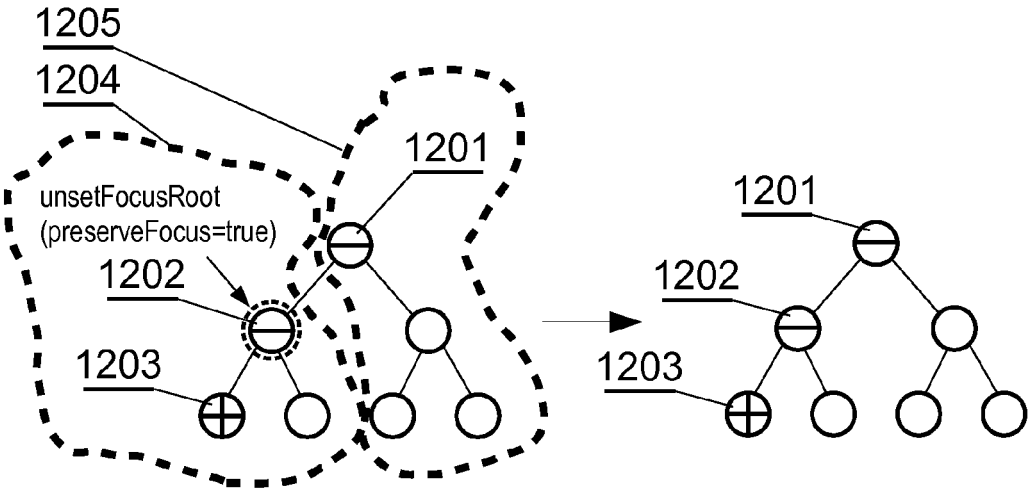


Fig. 12A

Fig. 12B

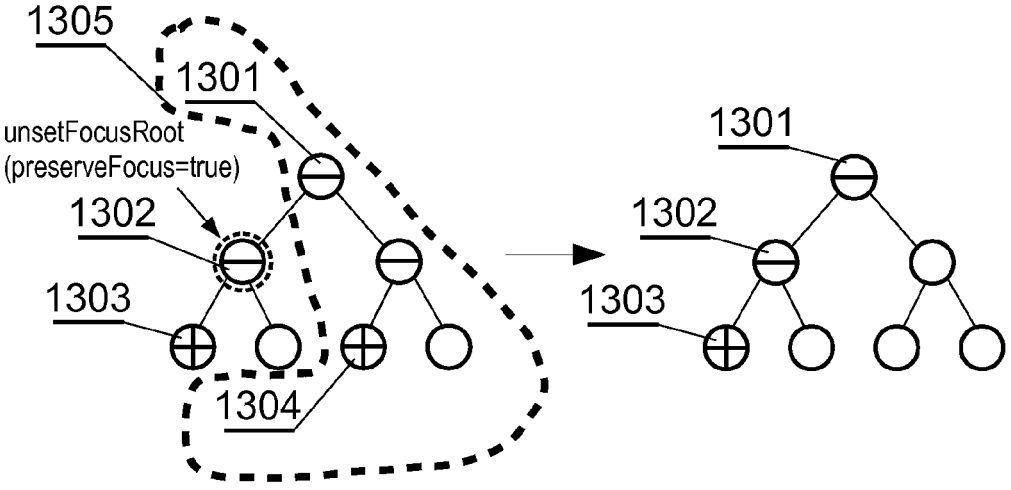


Fig. 13A

Fig. 13B

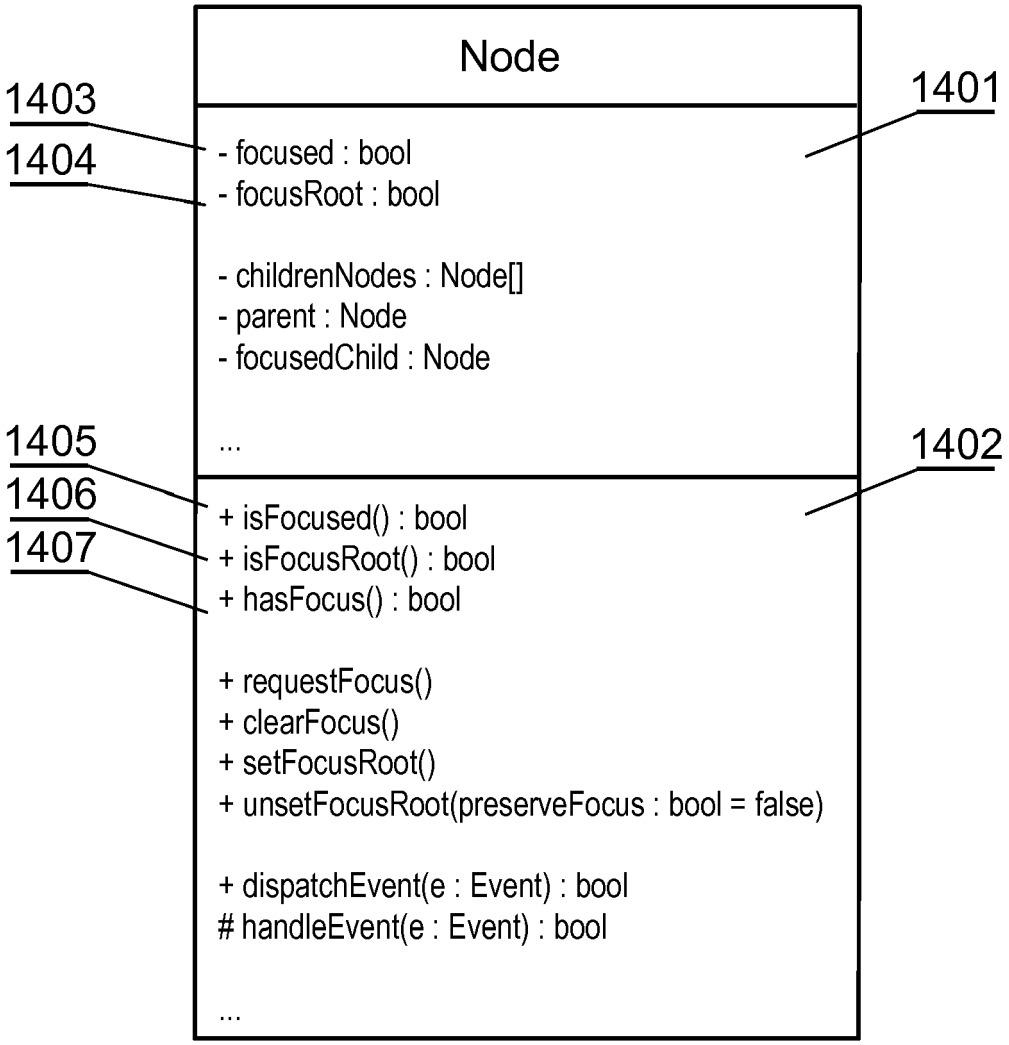
Node
1401
1403
1404
- focused : bool
- focusRoot : bool
- childrenNodes : Node[]
- parent : Node
- focusedChild : Node
...
1405
1406
1407
1402
+ isFocused() : bool
+ isFocusRoot() : bool
+ hasFocus() : bool
+ requestFocus()
+ clearFocus()
+ setFocusRoot()
+ unsetFocusRoot(preserveFocus : bool = false)
+ dispatchEvent(e : Event) : bool
# handleEvent(e : Event) : bool
...

Fig. 14

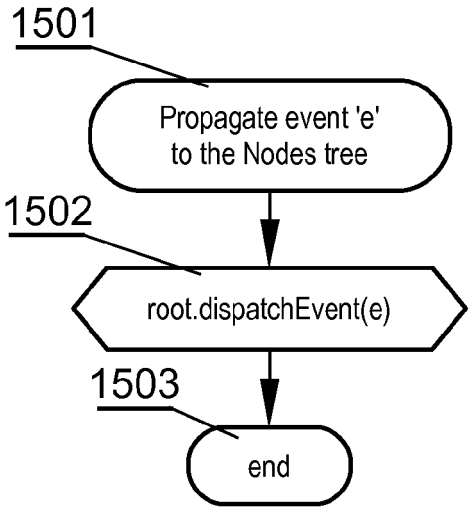
1501
Propagate event 'e'
to the Nodes tree
1502
root.dispatchEvent(e)
1503
end

Fig. 15

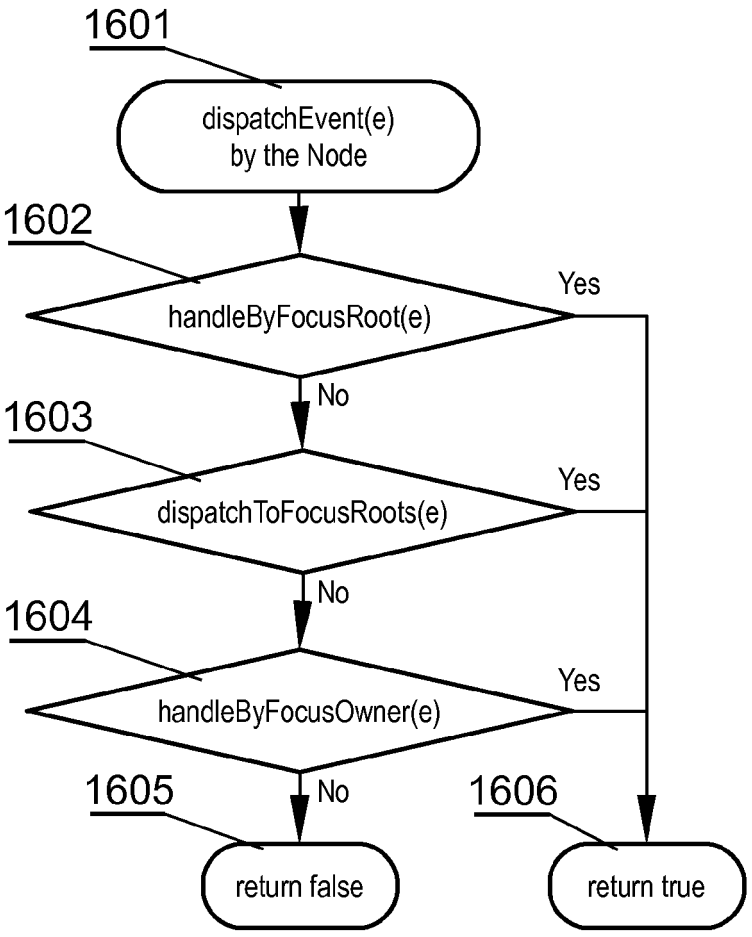
1601
dispatchEvent(e)
by the Node
1602
handleByFocusRoot(e)
Yes
No
1603
dispatchToFocusRoots(e)
Yes
No
1604
handleByFocusOwner(e)
Yes
No
1605
return false
1606
return true

Fig. 16

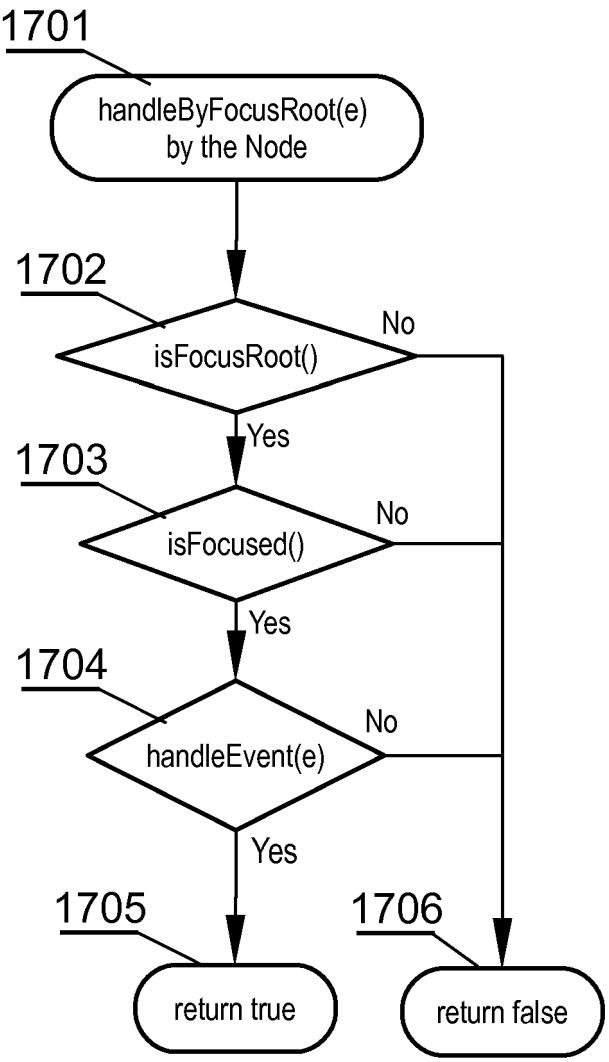
1701
handleByFocusRoot(e)
by the Node
1702
isFocusRoot()
No
Yes
1703
isFocused()
No
Yes
1704
handleEvent(e)
No
Yes
1705
return true
1706
return false

Fig. 17

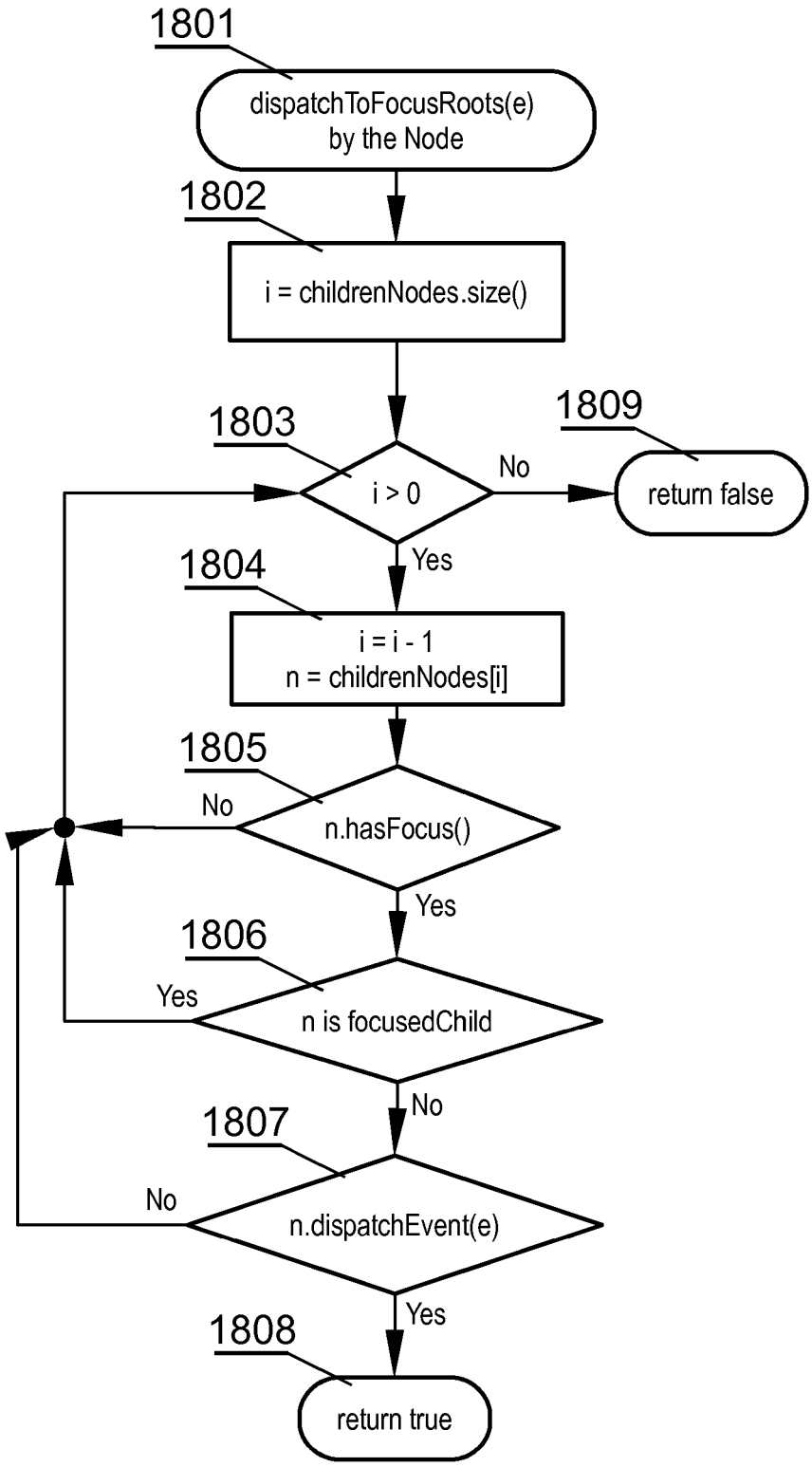
1801
dispatchToFocusRoots(e)
by the Node
1802
i = childrenNodes.size()
1803
i > 0
No
1809
return false
Yes
1804
i = i - 1
n = childrenNodes[i]
1805
No
n.hasFocus()
Yes
1806
Yes
n is focusedChild
No
1807
No
n.dispatchEvent(e)
Yes
1808
return true

Fig. 18

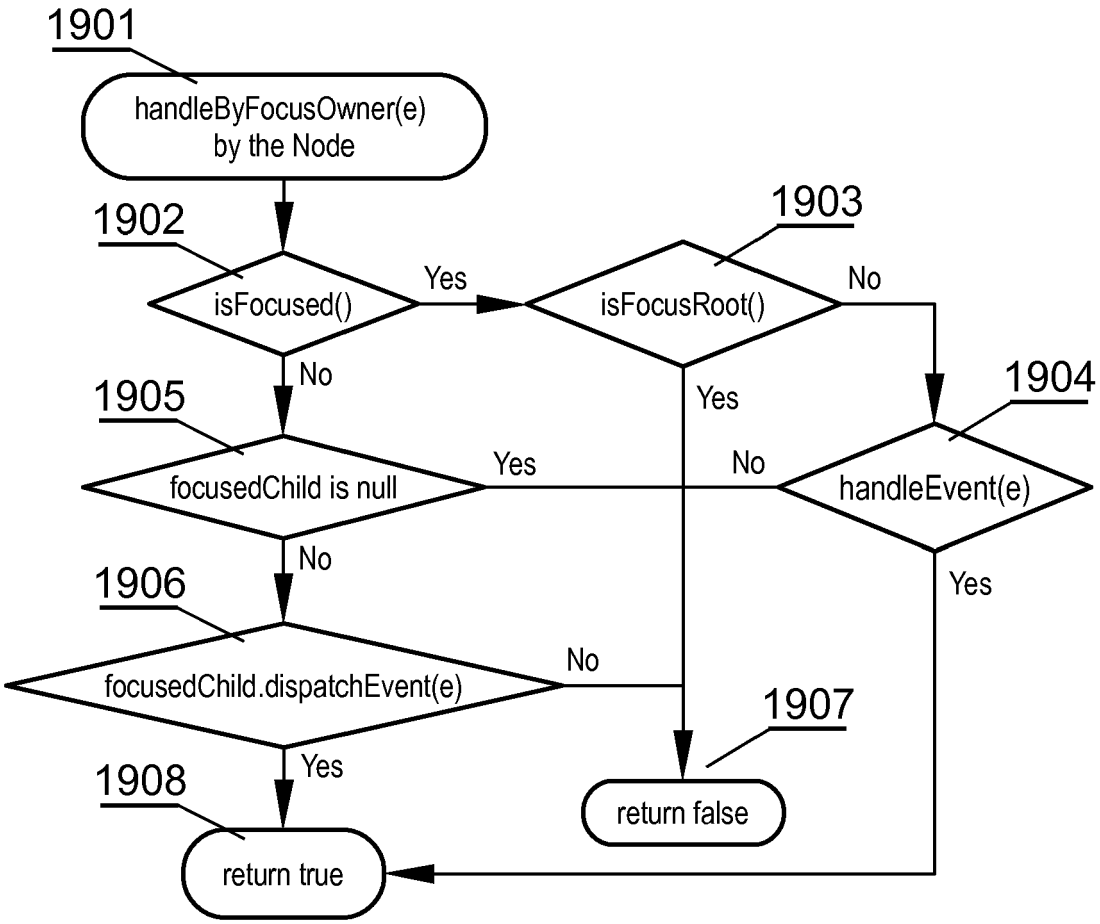
1901
handleByFocusOwner(e)
by the Node
1902
isFocused()
Yes
1903
isFocusRoot()
No
No
Yes
1904
handleEvent(e)
1905
focusedChild is null
Yes
No
No
1906
focusedChild.dispatchEvent(e)
No
1907
return false
Yes
1908
return true
Yes

Fig. 19

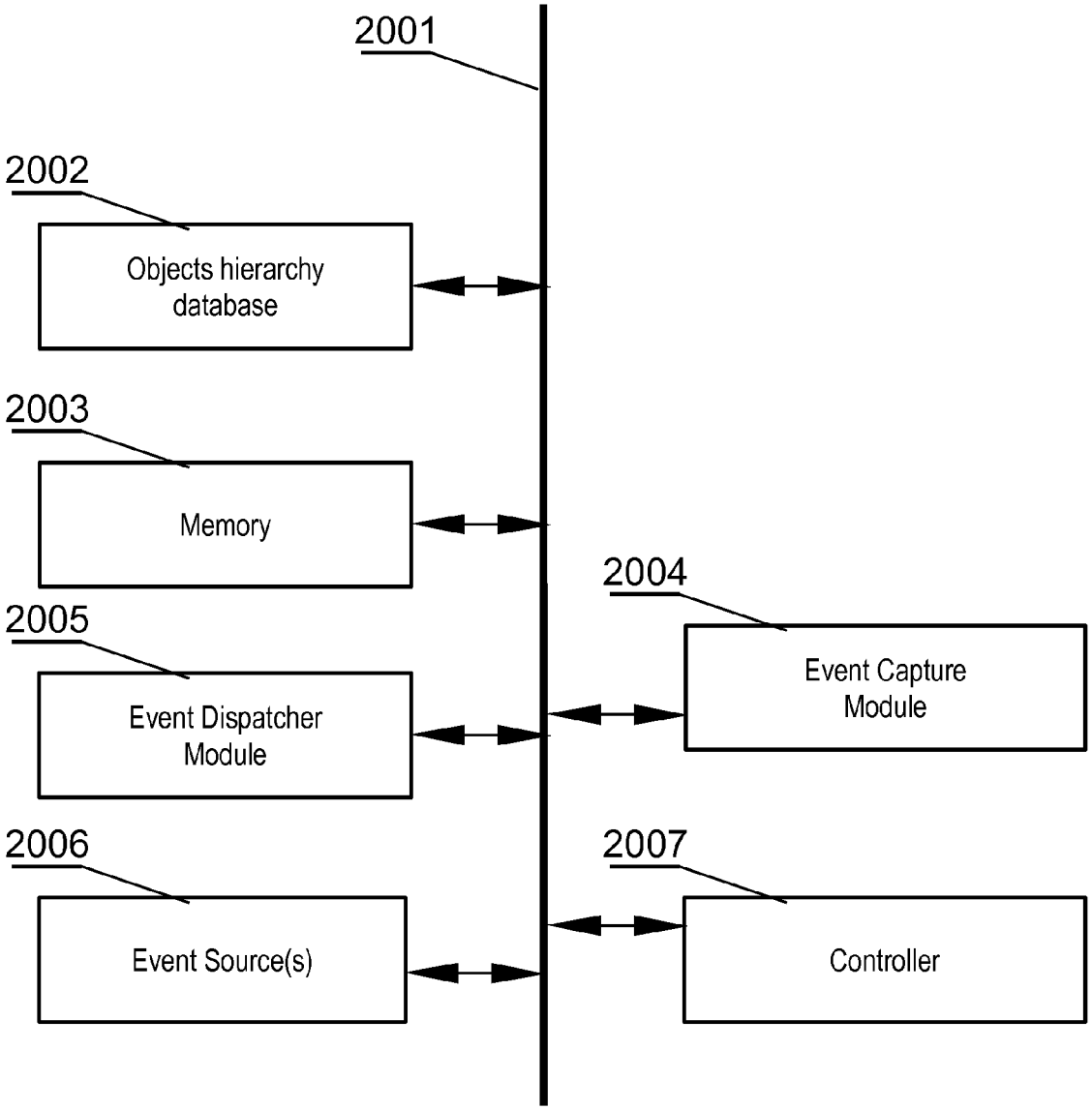
2001
2002
Objects hierarchy database
2003
Memory
2004
Event Capture Module
2005
Event Dispatcher Module
2006
Event Source(s)
2007
Controller

Fig. 20

Europäisches Patentamt
European Patent Office
Office européen des brevets

# EUROPEAN SEARCH REPORT

Application Number
EP 14 16 7608

## DOCUMENTS CONSIDERED TO BE RELEVANT

| Category | Citation of document with indication, where appropriate, of relevant passages | Relevant to claim | CLASSIFICATION OF THE APPLICATION (IPC) |
|---|---|---|---|
| X | ANONYMOUS: "public class View", , 9 December 2013 (2013-12-09), pages 1-18, XP002741498, Retrieved from the Internet: URL:http://developer.android.com/reference/android/view/View.html [retrieved on 2015-06-26] * page 1, "Class Overview", "Using Views" * * page 17, methods hasFocus(), isFocused() * | 1-15 | INV. G06F9/44 G06F3/0481 |
| T | ANONYMOUS: "UI Overview", , 17 October 2013 (2013-10-17), pages 1-2, XP002741499, Retrieved from the Internet: URL:http://developer.android.com/guide/topics/ui/overview.html [retrieved on 2015-06-26] * figure 1 * * page 1, "User Interface Layout" * | 1-15 | TECHNICAL FIELDS SEARCHED (IPC) G06F |
| A | US 2009/249339 A1 (LARSSON JOSEF [US] ET AL) 1 October 2009 (2009-10-01) * abstract * * figure 2 * * page 2, paragraph [0017] * | 1-15 | |
| A | EP 0 368 779 A2 (IBM [US]) 16 May 1990 (1990-05-16) * abstract * * figures 1,4 * * column 6, lines 27-31 * | 1-15 | |

The present search report has been drawn up for all claims

| Place of search | Date of completion of the search | Examiner |
|---|---|---|
| Munich | 3 July 2015 | Leineweber, Harald |

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

## ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 14 16 7608

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

03-07-2015

| Patent document cited in search report | Publication date | Patent family member(s) | Publication date |
|---|---|---|---|
| US 2009249339 A1 | 01-10-2009 | AU 2009232301 A1 | 08-10-2009 |
| | | CN 101981591 A | 23-02-2011 |
| | | EP 2260460 A1 | 15-12-2010 |
| | | JP 2011516969 A | 26-05-2011 |
| | | KR 20110000569 A | 03-01-2011 |
| | | RU 2010140069 A | 10-04-2012 |
| | | US 2009249339 A1 | 01-10-2009 |
| | | WO 2009123801 A1 | 08-10-2009 |
| EP 0368779 A2 | 16-05-1990 | BR 8905668 A | 05-06-1990 |
| | | CA 1318039 C | 18-05-1993 |
| | | DE 68927473 D1 | 02-01-1997 |
| | | EP 0368779 A2 | 16-05-1990 |
| | | JP H02130628 A | 18-05-1990 |
| | | US 4975690 A | 04-12-1990 |

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

## REFERENCES CITED IN THE DESCRIPTION

*This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.*

### Patent documents cited in the description


- US 8347215 B2 **[0008]**
- EP 0368779 B1 **[0010]**